Information and Computation 209 (2011) 108–117

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Coinduction for preordered algebra

Răzvan Diaconescu

Institute of Mathematics "Simion Stoilow" of the Romanian Academy, Bucureşti, Calea Griviţei 21, Romania

ARTICLE INFO

Article history:
Received 23 June 2008
Revised 29 April 2010
Available online 16 November 2010

Keywords:
Coinduction
Behavioural specification
Preordered algebra
Hidden algebra
Rewriting logic
Heterogeneous specification
CafeOBJ

ABSTRACT

We develop a combination, called *hidden preordered algebra*, between *preordered algebra*, which is an algebraic framework supporting specification and reasoning about transitions, and *hidden algebra*, which is the algebraic framework for behavioural specification. This combination arises naturally within the heterogeneous framework of the modern formal specification language CafeOBJ. The novel specification concept arising from this combination, and which constitutes its single unique feature, is that of *behavioural transition*. We extend the coinduction proof method for behavioural equivalence to coinduction for proving behavioural transitions.

1. Introduction

Modern algebraic specification practice and theory has extended the traditional many-sorted algebra based specification to several new paradigms. Two of the most promising ones are behavioural specification [4,8–10,13–15] and rewriting logic [2]. An important effort has been undertaken to develop languages and systems supporting such extensions of traditional algebraic specification. To mention just a couple of them, Maude [1] in the area of rewriting logic and CafeOBJ [3,5] in the area of behavioural specification and rewriting logic. The latter realizes both paradigms above mentioned within a single specification framework, which constitutes one of the earliest examples of a *heterogenous specification* language. However it has to be said that CafeOBJ employs a rather diluted but quite effective form of rewriting logic, which corresponds to the unlabelled form of rewriting logic, which we call *preordered algebra*.

In such heterogenous specification frameworks it is crucial that any two logical formalisms involved have a 'least upper bound', which should appear as a 'super-logic' to both of them. Thus in the case of a system like CafeOBJ one needs to study such a combination between hidden algebra (*HA*), i.e. the logic underlying the behavioural specification paradigm, and preordered algebra (*POA*). This has already been defined in model theoretic terms in [3,5] (although only the latter reference constitutes the definitive solution to this combination problem). The single characteristic outcome of this combination is the novel concept of behavioural transition. Although behavioural transitions already constitute a language construct in CafeOBJ, unfortunately its methodological aspects remain unexplored. Our current paper takes a first step to filling this gap by providing a coinduction-like proof method for behavioural transitions which extends the well known coinduction for proving behavioural equivalences.

Our paper is organised as follows:

1. We briefly review some basic concepts from many sorted algebra, hidden algebra, and preordered algebra needed for our work. Most notably, in this preliminary section we introduce a novel concept of congruence for preordered algebras.

E-mail address: Razvan.Diaconescu@imar.ro

0890-5401/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2010.11.001

2. In the next section we develop hidden preordered algebra (HPOA) and prove the main result of this paper, namely the existence of the largest hidden preordered congruence for any algebra.
3. In the final technical section we extract a coinduction principle for behavioural transitions and give an application as example.

Some of our examples are written in CafeOBJ. However the reader is not required to have deep knowledge of this notation since this follows anyway quite closely the mathematical notations for the basic algebraic specification concepts. When necessary we also provide additional explanations. The relationship between the terminology employed by our paper and that of the foundational work on CafeOBJ (as appears in [5]) is as follows. The logics RWL and HRWL, respectively, of the CafeOBJ cube of [5] appear here as POA and HPOA, respectively. For the sake of simplicity of presentation here we have not developed the order-sorted algebra dimension of the CafeOBJ cube (i.e. the *OS* logics). This aspect can be added easily to our framework. Note also that here we take the perspective of logic embedding, which is dual but in our case semantically equivalent to the perspective of logic projections that underlies the CafeOBJ cube in [5]. This simply means that from the perspective of the terminology of our paper the CafeOBJ cube of [5] should be read with all arrows reversed.

2. Preliminaries

2.1. Many-sorted algebra

This is the traditional framework for algebraic specification and constitutes the core framework for all algebraic specification formalisms. In the following we introduce the main concepts of many-sorted algebra needed by our work.

Definition 2.1 (*Many-sorted signatures*)**.** We let S^* denote the set of all finite sequences of elements from S, with [] the empty sequence. A(n S-*sorted*) *signature* (S, F) is an $S^* \times S$-indexed set $F = \{F_{w\to s} | w \in S^*,\ s \in S\}$ of *operation symbols*.

Note that this definition permits *overloading*, in that the sets $F_{w\to s}$ need *not* be disjoint. Call $\sigma \in F_{[]\to s}$ (sometimes denoted simply $F_{\to s}$) a *constant symbol* of sort s.

An (S, F)-*term* t of sort $s \in S$, is a structure of the form $\sigma(t_1, \ldots, t_n)$, where $\sigma \in F_{w\to s}$ and $t_1, \ldots, t_n$ are (S, F)-terms of sorts $s_1, \ldots, s_n$, where $w = s_1, \ldots, s_n$.

A *signature morphism* φ from a signature (S, F) to a signature (S', F') is a pair $(\varphi^{\text{sort}}, \varphi^{\text{op}})$ consisting of

- a map $\varphi^{\text{sort}} :\ S \to S'$ of sorts and
- maps $\varphi^{\text{op}}_{w\to s} :\ F_{w\to s} \to F'_{(\varphi^{\text{sort}})^*(w)\to\varphi^{\text{sort}}(s)}$ for all $w \in S^*$ and $s \in S$.

Definition 2.2 (*Many-sorted algebras*)**.** Given a *sort set* S, an S-*indexed* (or *sorted*) *set* A is a family $\{A_s\}_{s\in S}$ of sets indexed by the elements of S; in this context, $a \in A$ means that $a \in A_s$ for some $s \in S$. Given an S-indexed set A and $w = s_1, \ldots, s_n \in S^*$, we let $A_w = A_{s_1} \times \cdots \times A_{s_n}$; in particular, we let $A_{[]} = \{\bigstar\}$, some one point set.

A (S, F)-*algebra* A consists of

- an S-indexed set A (the set A_s is called the *carrier* of A of sort s) and
- a function $A_\sigma :\ A_w \to A_s$ for each $\sigma \in F_{w\to s}$.

If $\sigma \in F_{\to s}$ then A_σ determines a point in A_s which may also be denoted A_σ.

Given a signature morphism $\varphi :\ (S, F) \to (S', F')$ and a (S', F')-algebra A', we can define the φ-*reduct* of A' to (S, F) to be the following (S, F)-algebra:

- $A_s = A'_{\varphi^{\text{st}}(s)}$ for $s \in S$ and
- $A_\sigma = A'_{\varphi^{\text{op}}(\sigma)}$ for $\sigma \in F$.

The (S, F)-algebra A may be denoted $A'|_\varphi$ (or simply $A'|_{(S,F)}$ when φ is an inclusion of signatures). Then A' is called an φ-*expansion* of A along φ.

Definition 2.3 (*Algebra homomorphisms*)**.** An S-*indexed* (or *sorted*) *function* $f :\ A \to B$ is a family $\{f_s :\ A_s \to B_s\}_{s\in S}$.

Also, for an S-sorted function $f :\ A \to B$, we let $f_w :\ A_w \to B_w$ denote the product function mapping a tuple of elements $(a_1, \ldots, a_n)$ to the tuple $(f_{s_1}(a_1), \ldots, f_{s_n}(a_n))$.

An (S, F)-*homomorphism* from a (S, F)-algebra A to another (S, F)-algebra B is an S-indexed function $h :\ A \to B$ such that for each $\sigma \in F_{w\to s}$ and $a \in A_w$

$$h_s(A_\sigma(a)) = B_\sigma(h_w(a)).$$

Definition 2.4 (*Sentences*)**.** A (S, F)-*equation* is an equality $t = t'$ between (S, F)-terms t and t'. Equations are the simplest (S, F)-sentences.

For ρ_1 and ρ_2 any (S, F)-sentences, let $\rho_1 \wedge \rho_2$ be their conjunction which is also a (S, F)-sentence. Other Boolean connectives are disjunction ($\vee$), implication ($\Rightarrow$), negation ($\neg$), etc.

For any set X of variables for a signature (S, F), $(\forall X)\rho$ is an (S, F)-sentence for each $(S, F \cup X)$-sentence ρ. Likewise we have existential quantification.

Definition 2.5 (*Satisfaction*)**.** Any (S, F)-*term* $t = \sigma(t_1, \ldots, t_n)$, where $\sigma \in F_{w \to s}$ is an operation symbol and $t_1, \ldots, t_n$ are (S, F)-(sub)terms corresponding to the arity w, gets *interpreted as an element* $A_t \in A_s$ in a (S, F)-algebra A by $A_t = A_\sigma(A_{t_1}, \ldots, A_{t_n})$.

The *satisfaction relation* between algebras and sentences is the Tarskian satisfaction defined inductively on the structure of sentences. Given a fixed arbitrary signature (S, F) and an (S, F)-algebra A,

- $A \models t = t'$ if $A_t = A_{t'}$ for equations,
- $A \models \rho_1 \wedge \rho_2$ if $A \models \rho_1$ and $A \models \rho_2$ and similarly for the other Boolean connectives, and
- for each $(S, F \cup X)$-sentence $A \models (\forall X)\rho$ if $A' \models \rho$ for each expansion A' of A along the signature inclusion $(S, F) \hookrightarrow (S, F \cup X)$.

Definition 2.6 (*Congruences*)**.** An (S, F)-*congruence* on a (S, F)-algebra A is an S-sorted family of relations, $\equiv_s$ on A_s, each of which is an equivalence relation, and which also satisfy the *congruence property*, that given any $\sigma \in F_{w \to s}$ and any $a \in A_w$, then $A_\sigma(a) \equiv_s A_\sigma(a')$ whenever $a \equiv_w a'$.[1]

Definition 2.7. Each congruence on an (S, F)-algebra A determines a *quotient* algebra $A/_\equiv$ such that

- $(A/_\equiv)_s = (A_s)/_{\equiv_s}$ for each sort $s \in S$, i.e. the equivalence classes of $\equiv_s$ and
- $(A/_\equiv)_\sigma(a/_\equiv) = A_\sigma(a)/_\equiv$ for each operation symbol $\sigma \in F_{w \to s}$ and each $a \in A_w$.

Let the *kernel* $=_h$ of a homomorphism $h : A \to B$ be defined by $a =_h b$ if and only if $h(a) = h(b)$.

Fact 2.1. *For any algebra homomorphism h, its kernel* $=_h$ *is a congruence.*

2.2. Hidden algebra

This is the mathematical framework underlying the so-called 'behavioural specification' paradigm [4,8–10,13–15] which is a generalisation of ordinary (many-sorted) algebraic specification. Behavioural specification characterises how objects (and systems) *behave*, not how they are implemented. This new form of abstraction can be very powerful in the specification and verification of software systems since it naturally embeds other useful paradigms such as concurrency, object-orientation, constraints, nondeterminism, etc. (see [9] for details). Behavioural abstraction is achieved by using specification with hidden sorts and a behavioural concept of satisfaction based on the idea of indistinguishability of states that are observationally the same, which also generalises process algebra and transition systems (see [9]).

Our brief presentation of the main concepts of hidden algebra (abbreviated HA) given below follows the so-called 'coherent hidden algebra' [4,5] framework employed by CafeOBJ. This is both a simplification and extension of the classical hidden algebra of [8,9] in several directions, most notably by allowing operations with multiple hidden sorts in the arity, and differs only slightly from other modern formalizations of hidden algebra in the literature [10,15]. HA also is significantly more general than coalgebra with final semantics [11] since it integrates smoothly data types and it allows behavioural operations with multiple hidden sorts.

Definition 2.8 (*Hidden algebra signatures*)**.** A *hidden algebraic signature* $(H, V, F, F^{\mathrm{b}})$ consists of

- disjoint sets H of *hidden sorts*, V of (ordinary) *visible sorts*,
- an indexed family F of $(H \cup V)$-sorted operation symbols such that $(H \cup V, F)$ is a many-sorted signature, and
- a distinguished subset $F^{\mathrm{b}}_{w \to s} \subseteq F_{w \to s}$ of *behavioural operations* for each arity w and sort s such that w contains at least one hidden sort.

The 'hidden' sorts are used to specify the spaces of the states of objects (or abstract machines) while the 'visible' ones are used for the ordinary data types.

Definition 2.9 (*Hidden algebras*)**.** Given a hidden algebraic signature $(H, V, F, F^{\mathrm{b}})$, a $(H, V, F, F^{\mathrm{b}})$-*algebra* is just an $(H \cup V, F)$-algebra.

[1] Meaning $a_i \equiv_{s_i} a'_i$ for $i = 1, \ldots, n$, where $w = s_1, \ldots, s_n$ and $a = (a_1, \ldots, a_n)$.

Definition 2.10 (*Hidden congruence*)**.** Given a $(H, V, F, F^{\mathrm{b}})$-algebra A, a *hidden* $(H, V, F, F^{\mathrm{b}})$-congruence $\sim$ on A is just an $(H \cup V, F^{\mathrm{b}})$-congruence which is identity on the visible sorts.

Definition 2.11 (*Behavioural equivalence*)**.** The largest hidden $(H, V, F, F^{\mathrm{b}})$-congruence $\sim_A$ on a $(H, V, F, F^{\mathrm{b}})$-algebra A is called the *behavioural equivalence on A*.

A proof of the following crucial result can be found for example in [15].

Theorem 2.1. *Behavioural equivalence exists for any* $(H, V, F, F^{\mathrm{b}})$*-algebra.*

This result generalises the final semantics employed by the early hidden algebra frameworks [8] or by the coalgebraic approaches [11] to the situation of behavioural operations with multiple hidden sorts in the arity and of loose interpretation of the visible part of the signature.

Definition 2.12 (*HA sentences*)**.** Given a hidden algebraic signature $(H, V, F, F^{\mathrm{b}})$, a *behavioural equation* $t \sim t'$ consists of a pair of $(H \cup V, F)$-terms of the same sort.

The full set of sentences for the signature is obtained in the manner of Definition 2.4 from the (strict) equations $t = t'$ and from the behavioural equations $t \sim t'$ by iterative applications of Boolean connectives (conjunction, disjunction, negation, implication, etc.) and by universal and existential quantifications.

Definition 2.13 (*Behavioural satisfaction*)**.** An $(H, V, F, F^{\mathrm{b}})$-algebra A satisfies a behavioural equation $t \sim t'$, i.e. $A \models t \sim t'$, when $A_t \sim_A A_{t'}$. The satisfaction of all *HA* sentences by algebras is defined inductively on the structure of the sentences as in Definition 2.4.

The following simple example shows the difference between the satisfaction of (ordinary strict) $(H \cup V, F)$-equations $t = t'$ and that of behavioural $(H, V, F, F^{\mathrm{b}})$-equations $t \sim t'$.

Example 2.1. Consider the following CafeOBJ specification of a counter abstract machine.

```
mod* COUNTER {
  protecting(NAT)
  *[ Counter ]*
  bop add  : Nat Counter -> Counter
  bop read : Counter -> Nat
  var N : Nat
  var C : Counter
  eq read(add(N,C)) = N + read(C) .
}
```

The declaration `protecting(NAT)` represents an import of the data type of the natural numbers which does not alter them in any way (i.e. all algebras of `COUNTER` interpret the `NAT` part in a fixed way, as the standard model of the natural numbers), the specification has only one hidden sort `Counter`, and two behavioural operations `add` and `read`. The only equation of the specification reads as

$$(\forall N : \mathtt{Nat})(\forall C : \mathtt{Counter})\ \mathtt{read}(\mathtt{add}(N, C)) = N + \mathtt{read}(C).$$

It is rather easy to show that for any algebra A of `COUNTER` we have that

$$a \sim_A a' \text{ if and only if } A_{\mathtt{read}}(a) = A_{\mathtt{read}}(a').$$

Now let us consider the equation

$$\mathtt{add}(1, \mathtt{add}(2, c)) = \mathtt{add}(3, c).$$

This is satisfied in the strict sense by the algebra of `COUNTER` which interprets the sort `Counter` as the natural numbers and `add` as addition of natural numbers. However it is not satisfied by algebras for which `Counter` stores states having more refined information. Such an example is given by a 'history' algebra A which interprets `Counter` as the set of pairs of naturals $\langle m, n\rangle$, such that $A_{\mathtt{read}}(\langle m, n\rangle) = m$ and $A_{\mathtt{add}}(x, \langle m, n\rangle) = \langle x + m, n + 1\rangle$.

But the above equation is satisfied *behaviourally* by *any* algebra of `COUNTER` (the proof of this fact is rather easy and is left to the reader).

2.2.1. Coinduction

Theorem 2.1 provides the foundation for the rather famous coinduction proof method. Suppose that one wants to prove that two states, represented as terms s and s', are behaviourally equivalent. Then it is enough to perform the following steps:

1. Define an equivalence relation R (called a *coinduction relation*) for each hidden sort;
2. Prove that R is a hidden congruence; and
3. Prove that $s\ R\ s'$.

The coinduction proof method contains a heuristic component which is represented by the choice of the relation R. Often R happens to be the behavioural equivalence, however the coinduction method does not require this. The choice of R is thus left to the user which has to rely upon his insight into the problem. Some methods have been invented in order to assist and ease the process of finding such coinduction relations, such as the so-called 'circular coinduction' of [15].

2.3. Preordered algebra

This specification formalism (abbreviated *POA*) can be regarded as a simplified form of *rewriting logic* [12] which is a non-trivial extension of traditional algebraic specification towards specification and formal verification of concurrent systems. *POA* thus incorporates many different models of concurrency in a natural, simple, and elegant way. Due to these attributes *POA* has been adopted by the CafeOBJ language as a framework for specification and reasoning about transitions in general [5], with applications to algorithm specification and verification [3,6], to automatic generation of case analysis [3], etc.

The main difference between *POA* and the full version of rewriting logic [2] lies in the fact that *POA* does not permit full reasoning about multiple transitions between states (or system configurations), but provides proof support for reasoning about the *existence* of transitions between states (or configurations). At the level of the semantics, this amounts to the fact that the *POA* models are preorders rather than categories. This avoids many of the semantical complications resulting from the labelled version of rewriting logic of [2]. However, this simplification given by *POA* still allows a multitude of pragmatic methodologies and moreover has the advantage of a great semantical and methodological simplicity.

In the following we present the basic concepts of the *POA* framework.

Definition 2.14 (*POA signatures*)**.** The *POA* signatures are just the many-sorted signatures (of Definition 2.1).

Definition 2.15 (*Preordered algebras)***.** A *preordered algebra* $(M, \leq)$ for a signature (S, F) consists of an (S, F)-algebra M and of a family $\leq = \{\leq_s \subseteq M_s \times M_s | s \in S\}$ of preorders such that the interpretation of each operation in F is monotonic with respect to $\leq$.

Definition 2.16 (*POA homomorphism*)**.** A *homomorphism of preordered algebras* $h : (M, \leq) \to (N, \leq')$ is an algebra homomorphism $M \to N$ which is also monotonic with respect to the preorders $\leq$ and $\leq'$.

Definition 2.17 (*POA sentences*)**.** The *POA* atomic sentences are either equations $t = t'$ or *transitions* $t \twoheadrightarrow t'$ with t and t' being terms of the same sort. The *POA* sentences are formed from (atomic) equations and transitions by iterations of the usual logical connectives and quantification.

Definition 2.18 (*Satisfaction in POA*)**.** A transition $t \twoheadrightarrow t'$ is *satisfied* by a preordered algebra $(M, \leq)$ if and only if $M_t \leq M_{t'}$.

The following is an example of a *POA* specification in CafeOBJ.

Example 2.2. Consider the following specification of non-deterministic naturals. The following module specifies multi-sets of natural numbers.

```
mod! NNAT {
  extending(NAT)
  op _|_ : Nat Nat -> Nat { assoc comm }
  vars M N1 N2 : Nat
  eq M + (N1 | N2) = (M + N1) | (M + N2) .
}
```

The `extending` importation for the import of the data type of natural numbers (with addition $+$) just means that no natural numbers are collapsed but new 'non-deterministic' naturals are introduced to the standard model of natural numbers. The declaration `mod!` means that the only models considered are those which are *initial*. The only explicit equation guarantees that these initial models have indeed only multi-sets of naturals rather than expressions containing also `+`. The fact that this gives multi-sets of naturals is also determined by the two implicit associativity and commutativity equations specified as attributes for the constructor `_|_`.

The next step is to define a non-deterministic choice on the non-deterministic naturals. In fact this gives sense to non-determinism for this example. This choice is non-confluent and is specified as transitions.

```
mod! NNAT-CHOICE
{
  protecting(NNAT)
  vars M N : Nat
  trans N | M => N .
  trans N | M => M .
}
```

The initial model of this specification has all non-deterministic naturals as elements, with the addition operation `+` (but it is of course possible to consider any of the other standard operations on the naturals), and with the preorder given by the

multi-set reduction. For example $(1 \mid 2 \mid 3) \leq (1 \mid 3)$. It is easy to see that addition $+$ is monotonic with respect to this preorder.

The existence of initial models for preordered algebra specification is guaranteed for the case when all the sentences of the specification are Horn sentences. The proof of this result (see [7] for example) is rather similar to the ordinary many-sorted algebra case by using the concept of preordered algebra congruence introduced below. Since the focus of our paper is rather different we skip here the proof of this result.

In the following we extend the concept of congruence from ordinary many-sorted algebra (see Definition 2.6) to preordered algebras.

Definition 2.19 (*Preordered algebra congruences*)**.** A *POA-congruence* (preordered algebra congruence) on a preordered algebra $(M, \leq)$ for a signature (S, F) is a pair $(\sim, \sqsubseteq)$ such that

- $\sim$ is an (S, F)-congruence on the (S, F)-algebra M,
- $\sqsubseteq$ is a (n S-sorted) preorder on M which contains $\leq$, i.e. $\leq \subseteq \sqsubseteq$, and which is compatible with the operations, and
- $a' \sim a$, $a \sqsubseteq b$, $b \sim b'$ implies $a' \sqsubseteq b'$ for all elements a, a', b, b' of M.

Congruences form a partial order under inclusion, i.e. $(\sim, \sqsubseteq) \subseteq (\sim', \sqsubseteq')$ if and only if $\sim \subseteq \sim'$ and $\sqsubseteq \subseteq \sqsubseteq'$.

Proposition 2.1. *Each POA-congruence on a preordered algebra $(M, \leq)$ determines a quotient preordered algebra homomorphism $(M, \leq) \to (M, \leq)/_{(\sim, \sqsubseteq)}$ such that*

- *the algebra underlying $(M, \leq)/_{(\sim, \sqsubseteq)}$ is the quotient algebra $M/\sim$,*
- *the preorder $\leq'$ on $(M, \leq)/_{(\sim, \sqsubseteq)}$ is defined by $m/\sim \leq' m'/\sim$ if and only if $m \sqsubseteq m'$.*

Proof. The definition of the preorder relation $\leq'$ is correct since it is independent of the choice of m and m'. Indeed, let $m \sim m_1$ and $m' \sim m'_1$. Then by the definition of the preorder congruences we have that $m \sqsubseteq m'$ if and only if $m_1 \sqsubseteq m'_1$. The fact that $\sqsubseteq$ is a preorder implies that $\leq'$ is a preorder.

In order to complete the argument that $(M, \leq)/_{(\sim, \sqsubseteq)}$ is a preordered algebra we have to show that the interpretations of the operations are monotonic with respect to the preorder $\leq'$. Let σ be any operation symbol and $(m_1, \ldots, m_n)$ and $(m'_1, \ldots, m'_n)$ appropriate lists of arguments for M_σ. We have to prove that

$$(M/\sim)_\sigma (m_1/\sim, \ldots, m_n/\sim) \leq' (M/\sim)_\sigma (m'_1/\sim, \ldots, m'_n/\sim)$$

if $m_k/\sim \leq' m'_k/\sim$ for each $1 \leq k \leq n$. This is equivalent to

$$M_\sigma (m_1, \ldots, m_n)/\sim \leq' M_\sigma (m'_1, \ldots, m'_n)/\sim$$

and to

$$M_\sigma (m_1, \ldots, m_n) \sqsubseteq M_\sigma (m'_1, \ldots, m'_n).$$

The above relation follows because $m_k \sqsubseteq m'_k$ (since $m_k/\sim \leq' m'_k/\sim$) for each $1 \leq k \leq n$ and by the definition of the preordered congruence which guarantees that M_σ is monotonic with respect to the preorder $\sqsubseteq$. □

Definition 2.20. The *kernel* $ker(h)$ of a preordered algebra homomorphism $h : (M, \leq) \to (N, \leq')$ is a pair $(=_h, \leq_h)$ of binary relations such that $a =_h b$ if and only if $h(a) = h(b)$ and $a \leq_h b$ if and only if $h(a) \leq h(b)$.

Fact 2.2. *$ker(h)$ is a POA-congruence.*

3. Hidden preordered algebra

Specification frameworks which contain both *POA* and *HA*, such as CafeOBJ, require a combination of both of them which would emerge as a 'super-logic' to both *POA* and *HA*.

Example 3.1. Consider a specification of a counter (like in Example 2.1) but this time with non-deterministic naturals (see Example 2.2) instead of the ordinary naturals. This may be achieved just by replacing `NAT` with `NNAT-CHOICE` in `COUNTER`; let us denote this version of `COUNTER` that uses non-deterministic naturals rather than naturals by `NCOUNTER`. The semantics of this module can be given only in a framework which contains both *POA* (for `NNAT-CHOICE`) and *HA* (for `COUNTER`).

Hidden preordered algebra (abbreviated *HPOA*) defined below, gives a natural combination between *POA* and *HA*, both of them appearing as 'sub-logics' of *HPOA*.

Definition 3.1 (*HPOA signatures*)**.** The signatures of hidden preordered algebra are the HA signatures (Definition 2.8).

Definition 3.2 (*HPOA models*)**.** The models of a HPOA signature $(H, V, F, F^{\mathrm{b}})$ are the preordered $(H \cup V, F)$-algebras.

Definition 3.3 (*HPOA sentences*)**.** The sentences of a HPOA signature are formed from atomic (strict) equations $t = t'$, behavioural equations $t \sim t'$, transitions $t \to t'$, and *behavioural transitions* $t \sim\!> t'$ by iteration of Boolean connectives and of quantifiers.

The following couple of definitions are crucial contributions of this paper. While the former combines the concept of POA congruence (Definition 2.19) with the concept of hidden congruence (Definition 2.10), the latter constitutes a generalisation of the concept of behavioural equivalence.

Definition 3.4 (*Hidden POA congruence*)**.** A *hidden POA-congruence* on a HPOA-algebra $(A, \leq)$ is a POA $(H \cup V, F^{\mathrm{b}})$-congruence $(\equiv, \sqsubseteq)$ on $(A, \leq)$ such that on the visible sorts $\equiv$ is the identity and $\sqsubseteq$ is $\leq$.

Definition 3.5 (*Behavioural POA congruence*)**.** The *behavioural POA-congruence* on $(A, \leq)$ is the which is the largest hidden POA-congruence on $(A, \leq)$.

Definition 3.6 (*Satisfaction in HPOA*)**.** The satisfaction of HPOA sentences by HPOA models (algebras) is defined inductively on the structure of the sentences like in Definition 2.5, where the satisfaction of the atomic equations $t = t'$ is given by Definition 2.5, that of atomic transitions $t \to t'$ by Definition 2.18, that of the atomic behavioural equations $t \sim t'$ by Definition 2.13, and

$$(A, \leq) \models t \sim\!> t' \text{ if and only if } A_t \lesssim_A A_{t'}.$$

Note that HPOA is more than just putting POA and HA together because of the behavioural transitions. The concept of behavioural transition arises naturally by symmetry to that of behavioural equation. The CafeOBJ language supports the specification of behavioural transitions by the keyword `btrans`. The definition of HPOA above relies upon the existence of behavioural POA-congruence, a result which represents an extension of Theorem 2.1 from HA to HPOA and which is developed below.

Our definitions of HPOA corrects the corresponding definitions from [5] by defining the concepts of behavioural equivalence and behavioural POA congruence, respectively, as the largest hidden congruence and hidden POA congruence, respectively, rather than defining them with contexts as in [5]. When the visible (data) part of the algebras are not reachable the respective concepts of behavioural equivalence of behavioural POA congruence differ from our paper to [5].

Theorem 3.1. *Behavioural POA-congruence exists for any preordered $(H, V, F, F^{\mathrm{b}})$-algebra.*

Proof. Let $(A, \leq)$ be any preordered $(H, V, F, F^{\mathrm{b}})$-algebra. We extend the signature of $(A, \leq)$ by adding the elements of A as new constants; let F_A be the new set of operation symbols thus obtained. Then F_A may be formally defined as

$$(F_A)_{w \to s} = \begin{cases} F_{w \to s} & \text{when } w \neq [] \\ F_{\to s} \uplus A_s & \text{when } w = [] \end{cases}.$$

Then $(A, \leq)$ can be expanded canonically to a preordered $(H, V, F_A, F^{\mathrm{b}})$-algebra $(A_A, \leq)$ which interprets each new constant (i.e. element of A) as itself, that is $(A_A)_a = a$ for each $a \in A$.

Let z be a new constant of any sort in $H \cup V$. Any $(H \cup V, F_A \uplus \{z\})$-term c is a *behavioural context for A* when it is either

- z (which is a new constant), or
- $\sigma(t_1, \ldots, c', \ldots, t_n)$ where σ is a constant or a behavioural operation, c' is a behavioural context, and $t_1, \ldots, t_n$ are F_A-terms.

A behavioural context c is called *visible* when the sort of c belongs to V.

For any element $a \in A$ of the same sort as z, let $c(z/a)$ denote the $(H \cup V, F_A)$-term obtained from c by replacing z by a. Then for any elements a and a' of A, having the same sort, let us define

$$a \sim_A a' \text{ if and only if } (A_A)_{c(z/a)} = (A_A)_{c(z/a')} \text{ for all visible behavioural contexts } c$$

and

$$a \lesssim_A a' \text{ if and only if } (A_A)_{c(z/a)} \leq_A (A_A)_{c(z/a')} \text{ for all visible behavioural contexts } c.$$

We show that $(\sim_A, \lesssim_A)$ is the behavioural POA-congruence on $(A, \leq)$ by showing first that it is a hidden POA-congruence and then that it is the largest one.

By definition it is clear that $\sim_A$ is an equivalence. Let us also note that from the definition of $\sim_A$, if the sort of a and of a' is visible we obtain that $a \sim_A a'$ if and only if $a = a'$ by taking the context c to be just z. Now let us consider a behavioural operation $\sigma \in F^{\mathrm{b}}_{w \to s}$. For any appropriate lists of arguments for A_σ, namely $(a_1, \ldots, a_n)$ and $(a'_1, \ldots, a'_n)$, we have to show that

$$A_\sigma(a_1, \ldots, a_n) \sim_A A_\sigma(a'_1, \ldots, a'_n) \text{ if } a_k \sim_A a'_k \text{ for each } 1 \leq k \leq n.$$

We first show that this relation holds for the particular case when $a_2 = a'_2, a_3 = a'_3, \ldots, a_n = a'_n$. Let us consider any visible sorted behavioural context c. We build a new context c' by

$$c' = c(\sigma(z, a_2, \ldots, a_n)).$$

Since $(A_A)_{c(z/A_\sigma(a_1,a_2,\ldots,a_n))} = (A_A)_{c'(z/a_1)}$ and $(A_A)_{c(z/A_\sigma(a'_1,a_2,\ldots,a_n))} = (A_A)_{c'(z/a'_1)}$ and because $a_1 \sim_A a'_1$ we obtain that $(A_A)_{c(z/A_\sigma(a_1,a_2,\ldots,a_n))} = (A_A)_{c(z/A_\sigma(a'_1,a_2,\ldots,a_n))}$, hence

$$A_\sigma(a_1, a_2, \ldots, a_n) \sim_A A_\sigma(a'_1, a_2 \ldots, a_n).$$

Now by replicating the same argument for a_2 and a'_2 when $a'_1, a_3, \ldots, a_n$ are fixed, and then further for a_3 and a'_3 and so on, because the arity of σ is finite, by the transitivity of $\sim_A$, we finally get that

$$A_\sigma(a_1, \ldots, a_n) \sim_A A_\sigma(a'_1, \ldots, a'_n).$$

We have thus shown that $\sim_A$ is a hidden congruence.

That $\lesssim_A$ is reflexive and transitive follows directly from its definition. The proof that the behavioural operations of A are monotonic with respect to $\lesssim_A$ is similar to the proof that the behavioural operations of A preserve $\sim_A$. In order to complete the proof that $\lesssim_A$ is a hidden POA-congruence let us consider $a \lesssim_A a'$ and $a \sim_A a_1$ and $a' \sim_A a'_1$. Then for each visible behavioural context c we have that $(A_A)_{c(z/a)} \leq_A (A_A)_{c(z/a')}$ and $(A_A)_{c(z/a)} = (A_A)_{c(z/a_1)}$ and $(A_A)_{c(z/a')} = (A_A)_{c(z/a'_1)}$. This implies that $(A_A)_{c(z/a_1)} \leq (A_A)_{c(z/a'_1)}$ for each visible behavioural context c, which means that $a_1 \lesssim_A a'_1$.

We have thus shown that $(\sim_A, \lesssim_A)$ is a hidden POA-congruence. Now we have to prove that it is the largest one. For this we consider another hidden POA-congruence $(\equiv, \sqsubseteq)$ on $(A, \leq)$. For any a and a' we can prove by induction on the depth of c, that for any behavioural context c

1. if $a \equiv a'$ then (because the behavioural operations on A preserve $\equiv$) $(A_A)_{c(z/a)} \equiv (A_A)_{c(z/a')}$, and
2. if $a \sqsubseteq a'$ then (because the behavioural operations on A preserve $\sqsubseteq$) $(A_A)_{c(z/a)} \sqsubseteq (A_A)_{c(z/a')}$.

In particular if c is visible sorted the above relations says that $(A_A)_{c(z/a)} = (A_A)_{c(z/a')}$ and $(A_A)_{c(z/a)} \leq_A (A_A)_{c(z/a')}$, respectively. This just shows that $(\equiv, \sqsubseteq) \subseteq (\sim_A, \lesssim_A)$. □

Note that $\sim_A$ is just the behavioural equivalence on A when we forget about its preordered structure. Hence the proof of Theorem 3.1 above contains the proof of the existence of behavioural equivalences Theorem 2.1 as a special case.

4. Coinduction for hidden preorder algebras

From Theorem 3.1 we can extract the following coinduction proof principle for hidden preorder algebras.

Corollary 4.1 (Coinduction for behavioural transitions)**.** *The coinduction proof method for HPOA consists of the following steps:*

1. *Define an equivalence relation R and a preorder relation P for each hidden sort such that*

   $$(s\ P\ s') \text{ and } (s\ R\ s1) \text{ and } (s'\ R\ s'1) \text{ imply } (s1\ P\ s'1)$$

   and

   $$(s \twoheadrightarrow s') \text{ implies } (s\ P\ s')$$

   for all $s, s', s1, s'1$.
2. *Prove that both R and P are preserved by the behavioural operations.*
3. *(a) If we want to prove that* $t \sim t'$ *then we show that* $t\ R\ t'$*, and*
   (b) If we want to prove that $t \sim\!> t'$ *then we show that* $t\ P\ t'$.

Example 4.1. For the specification NCOUNTER let us show by coinduction that

$$\texttt{add}(p,\ \texttt{add}(m \mid n, c)) \sim\!> \texttt{add}(m,\ \texttt{add}(p,\ c))$$

for all non-deterministic naturals m, n, and p, and for each state c of Counter. Let us first note that this transition does not hold in the strict sense. One model M of NCOUNTER that does not satisfy

$$\texttt{add}(p,\ \texttt{add}(m \mid n, c)) \twoheadrightarrow \texttt{add}(m,\ \texttt{add}(p,\ c))$$

may be defined as follows:

- $(M_{\texttt{Counter}}, \leq)$ is the partial ordered set of the lists of non-deterministic naturals, i.e. lists of elements of $M_{\texttt{Nat}}$. The partial order between these lists is defined by $(m_1)(m_2)\ldots(m_k) \leq (n_1)(n_2)\ldots(n_j)$ iff $k = j$ and $m_i \leq n_i$ for each $i \in \{1, \ldots, k\}$. (Recall that the partial order $m \leq n$ on the non-deterministic naturals is given by the existence of a transition, a 'choice' in this case, from m to n.)
- $M_{\texttt{add}}(m, c) = (m)c$ for each $m \in M_{\texttt{Nat}}$ and $c \in M_{\texttt{Counter}}$.
- $M_{\texttt{read}}((m_1)\ldots(m_k)) = m_1 + \cdots + m_k$.

Note that since for example $(1)(2|3) \not\leq (2|1)$, the considered transition does not hold in the strict sense.

Proposition 4.1. $\texttt{COUNTER} \models \texttt{add}(p,\ \texttt{add}(m \mid n, c)) \sim\!> \texttt{add}(m,\ \texttt{add}(p,\ c))$ *for all m, n, and p of sort* Nat *and all c of sort* Counter.

Proof. Instead of a conventional mathematical proof we present a CafeOBJ proof score implementing the coinduction method for *HPOA* extracted above. The actual proof is performed by the reductions below (command reduce) which are done by the CafeOBJ system rewriting, and all of them give the answer true.

1. The following introduces the relations R and P required by the *HPOA* coinduction method.

```
mod* NCOUNTER-PROOF {
  protecting(NCOUNTER)
  op _R_ : Counter Counter -> Bool
  op _P_ : Counter Counter -> Bool
  vars C C' : Counter
  eq C R C' = read(C) ==  read(C') .
  eq C P C' = read(C) ==> read(C') .
}
```

(The predicates == and ==> are the CafeOBJ built-in semantic equality and preorder predicates, respectively.) Note the rather simple definitions for R and P.

The following is the proof score for the fact that

$$(\texttt{s}\ P\ \texttt{s'})\ \text{and}\ (\texttt{s}\ R\ \texttt{s1})\ \text{and}\ (\texttt{s'}\ R\ \texttt{s'1})\ \text{imply}\ (\texttt{s1}\ P\ \texttt{s'1}).$$

```
open NCOUNTER-PROOF .
ops s s' s1 s'1 s2 s'2 : -> Counter .
```

(By the couple of declarations above we have introduced new arbitrary temporary constants which play the role of variables. They cease to exist when the module is closed back by the command close.)

We now introduce the hypotheses:

```
trans read(s) => read(s') .
eq read(s1) = read(s) .
eq read(s'1) = read(s') .
```

and execute the conclusion:

```
red s1 P s'1 .
```

We now proceed with the proof that

$$(\texttt{s -> s'})\ \text{implies}\ (\texttt{s}\ P\ \texttt{s'}).$$

The following is the hypothesis:

```
trans s2 => s'2 .
```

and now we execute the conclusion:

```
 reduce s2 P s'2 .
close
```

2. The next step is to prove that both R and P are preserved by the behavioural operations. For the case of `read` this property holds by the definitions of R and P. We therefore focus on `add`.

```
open NCOUNTER-PROOF .
ops s s' s1 s'1 s2 s'2 : -> Counter .
op n : -> Nat .
```

We introduce the hypotheses:

```
eq read(s1) = read(s) .
trans read(s2) => read(s'2) .
```

and now we execute the conclusions:

```
 red add(n,s) R add(n,s1) .
 red add(n,s2) P add(n,s'2) .
close
```

3. The final step is the proof of the actual property of this proposition.

```
 open NCOUNTER-PROOF .
 ops m n p : -> Nat .
 op c : -> Counter .
reduce add(p, add(m | n, c)) P add(m, add(p, c)) .
close
```

□

5. Conclusions

We have defined an upper bound logic for *POA* and *HA* as a natural combination between them and based on the novel concept of *POA*-congruence introduced we have proved the existence of the largest behavioural *POA*-congruence for any hidden preordered algebra. From this result we have extracted a coinduction principle for hidden preordered algebras which subsumes the well known coinduction principle of hidden algebra but also provides a proof method for the novel concept of behavioural transition.

Future work needs to focus on developing pragmatic methodologies for using behavioural transitions in specification. This seems to us a very promising research direction which has not been explored yet.

Acknowledgments

The author thanks both anonymous referees for their comments that have helped improve the presentation of this work.

References

[1] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, Carolyn Talcott, All About Maude – A High-performance Logical Framework, Lecture Notes in Computer Science, vol. 4350, Springer, 2007.
[2] Manuel Clavel, Steve Eker, Patrick Lincoln, Jose Meseguer, Principles of Maude, in: Proceedings, First International Workshop on Rewriting Logic and its Applications, Asilomar, California, September 1996, Electronic Notes in Theoretical Computer Science, vol. 4, 1996.
[3] Răzvan Diaconescu, Kokichi Futatsugi, CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification, AMAST Series in Computing, vol. 6, World Scientific, 1998.
[4] Răzvan Diaconescu, Kokichi Futatsugi, Behavioural coherence in object-oriented algebraic specification, Universal Computer Science 6 (1) (2000) 4–96, First version appeared as JAIST Technical Report IS-RR-98-0017F, June 1998.
[5] Răzvan Diaconescu, Kokichi Futatsugi, Logical foundations of CafeOBJ, Theoretical Computer Science 285 (2002) 289–318.
[6] Răzvan Diaconescu, Kokichi Futatsugi, Shusaku Iida, CafeOBJ Jewels, in: Kokichi Futatsugi, Ataru Nakagawa, Tetsuo Tamai (Eds.), Cafe: An Industrial-strength Algebraic Formal Method, Elsevier, 2000.
[7] Răzvan Diaconescu, Institution-independent Model Theory, Birkhäuser, 2008.
[8] Joseph Goguen, Răzvan Diaconescu, Towards an algebraic semantics for the object paradigm, in: Harmut Ehrig, Fernando Orejas (Eds.), Recent Trends in Data Type Specification, Lecture Notes in Computer Science, vol. 785, Springer, 1994, pp. 1–34.
[9] Joseph Goguen, Grant Malcolm, A hidden agenda, Theoretical Computer Science 245 (1) (2000) 55–101.
[10] Rolf Hennicker, Michel Bidoit, Observational logic, in: A.M. Haeberer (Ed.), Algebraic Methodology and Software Technology, Proc. AMAST'99, Lecture Notes in Computer Science, vol. 1584, Springer, 1999, pp. 263–277.
[11] B. Jacobs, J.M. Rutten, A tutorial on (co)algebras and (co)induction, Bulletin of EATCS 62 (1997) 222–259.
[12] José Meseguer, Conditional rewriting logic as a unified model of concurrency, Theoretical Computer Science 96 (1) (1992) 73–155.
[13] Horst Reichel, Behavioural equivalence – a unifying concept for initial and final specifications, in: Proceedings, Third Hungarian Computer Science Conference, Akademiai Kiado, Budapest, 1981.
[14] Horst Reichel, Initial Computability, Algebraic Specifications, and Partial Algebras, Clarendon, 1987.
[15] Grigore Roşu, Hidden Logic, Ph.D. Thesis, University of California at San Diego, 2000.